Date=06/10/2020
Lecture By=Arkesh Jaiswal
Subject ⇒Javascript

IN PREVIOUS LECTURE **(QUICK RECAP) Date-05/10/2020**	**In Today's Lecture (Overview)**
JavaScript Conditional Statements The JavaScript Switch Statement The break Keyword JavaScript Loops Questions For Self Practice	• JavaScript Arrays • JavaScript Array Methods • Popping and Pushing • Pushing • Shifting Elements • Deleting Elements • Splicing an Array • Slicing an Array • Questions for self Practice / CC For The Day

JavaScript Arrays

JavaScript arrays are used to store multiple values in a single variable.

Example

```
var cars = ["Saab", "Volvo", "BMW"];
```

What is an Array?

An array is a special variable, which can hold more than one value at a time.

If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
var car1 = "Saab";
var car2 = "Volvo";
var car3 = "BMW";
```

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?

The solution is an array!

An array can hold many values under a single name, and you can access the values by referring to an index number.

Creating an Array

Using an array literal is the easiest way to create a JavaScript Array.

Syntax:

```
var array_name = [item1, item2, ...];
```

Example

```
var cars = ["Saab", "Volvo", "BMW"];
```

Creating an Array

Using an array literal is the easiest way to create a JavaScript Array.

Syntax:

```
var array_name = [item1, item2, ...];
```

Example

```
var cars = ["Saab", "Volvo", "BMW"];
```

Spaces and line breaks are not important. A declaration can span multiple lines:

Example

```
var cars = [
  "Saab",
  "Volvo",
  "BMW"
];
```

Array Constructor

You can initialize an array with Array constructor syntax using new keyword.

The Array constructor has following three forms.

Syntax:

```
var arrayName = new Array();

var arrayName = new Array(Number length);

var arrayName = new Array(element1, element2, element3,... elementN);
```

Accessing Array Elements

An array elements (values) can be accessed using index (key). Specify an index in square bracket with array name to access the element at particular index. Please note that index of an array starts from zero in JavaScript.

Example: Access Array Elements

```
var stringArray = new Array("one", "two", "three", "four");

stringArray[0]; // returns "one"

stringArray[1]; // returns "two"

stringArray[2]; // returns "three"

stringArray[3]; // returns "four"

var numericArray = [1, 2, 3, 4];

numericArray[0]; // returns 1

numericArray[1]; // returns 2

numericArray[2]; // returns 3

numericArray[3]; // returns 4
```

Array Properties

Array includes "length" property which returns number of elements in the array.

Use for loop to access all the elements of an array using length property.

Example: Access Array using for Loop

```
var stringArray = new Array("one", "two", "three", "four");
for (var i = 0; i < stringArray.length ; i++)
{
stringArray[i];
}
```

 Points to Remember :

1. An array is a special type of variable that stores multiple values using a special syntax.
2. An array can be created using array literal or Array constructor syntax.
3. Array literal syntax: `var stringArray = ["one", "two", "three"];`
4. Array constructor syntax:`var numericArray = new Array(3);`
5. A single array can store values of different data types.
6. An array elements (values) can be accessed using zero based index (key). e.g. array[0].
7. An array index must be numeric.
8. Array includes length property and various methods to operate on array objects.

JavaScript Array Methods

Popping and Pushing

When you work with arrays, it is easy to remove elements and add new elements.

This is what popping and pushing is:

Popping items out of an array, or pushing items into an array.

Popping

The `pop()` method removes the last element from an array:

Example

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];

fruits.pop();               // Removes the last element ("Mango") from
fruits
```

The `pop()` method returns the value that was "popped out":

Example

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];

var x = fruits.pop();       // the value of x is "Mango"
```

Pushing

The `push()` method adds a new element to an array (at the end):

Example

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];

fruits.push("Kiwi");        //  Adds a new element ("Kiwi") to fruits
```

The `push()` method returns the new array length:

Example

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];

var x = fruits.push("Kiwi");    //  the value of x is 5
```

Shifting Elements

Shifting is equivalent to popping, working on the first element instead of the last.

The `shift()` method removes the first array element and "shifts" all other elements to a lower index.

Example

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];

fruits.shift();            // Removes the first element "Banana" from
fruits
```

The `shift()` method returns the string that was "shifted out":

Example

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];

var x = fruits.shift();    // the value of x is "Banana"
```

The `unshift()` method adds a new element to an array (at the beginning), and "unshifts" older elements:

Example

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];

fruits.unshift("Lemon");    // Adds a new element "Lemon" to fruits
```

The `unshift()` method returns the new array length.

Example

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];

fruits.unshift("Lemon");    // Returns 5
```

Deleting Elements

Since JavaScript arrays are objects, elements can be deleted by using the JavaScript operator `delete`:

Example

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];

delete fruits[0];           // Changes the first element in fruits to
undefined
```

Using delete may leave undefined holes in the array. Use pop() or shift() instead.

Splicing an Array

The `splice()` method can be used to add new items to an array:

Example

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];

fruits.splice(2, 0, "Lemon", "Kiwi");
```

The first parameter (2) defines the position where new elements should be added (spliced in).

The second parameter (0) defines how many elements should be removed.

The rest of the parameters ("Lemon" , "Kiwi") define the new elements to be added.

The `splice()` method returns an array with the deleted items:

Example

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];

fruits.splice(2, 2, "Lemon", "Kiwi");
```

Using splice() to Remove Elements

With clever parameter setting, you can use `splice()` to remove elements without leaving "holes" in the array:

Example

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];

fruits.splice(0, 1);         // Removes the first element of fruits
```

The first parameter (0) defines the position where new elements should be added (spliced in).

The second parameter (1) defines how many elements should be removed.

The rest of the parameters are omitted. No new elements will be added.

Slicing an Array

The `slice()` method slices out a piece of an array into a new array.

This example slices out a part of an array starting from array element 1 ("Orange"):

Example

```
var fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];

var citrus = fruits.slice(1);
```

The `slice()` method creates a new array. It does not remove any elements from the source array.

This example slices out a part of an array starting from array element 3 ("Apple"):

Example

```
var fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];

var citrus = fruits.slice(3);
```

The `slice()` method can take two arguments like `slice(1, 3)`.

The method then selects elements from the start argument, and up to (but not including) the end argument.

Example

```
var fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];
```

```
var citrus = fruits.slice(1, 3);
```

Questions for self Practice / CC For The Day

https://au-assignment.s3.amazonaws.com/cc-1a0217c0-0a78-4865-9a94-e1185af6a556.pdf